(19) 대한민국특허청(KR)
(12) 등록특허공보(B1)

(45) 공고일자 2021년12월06일
(11) 등록번호 10-2336521
(24) 등록일자 2021년12월02일

(51) 국제특허분류(Int. Cl.)
G06F 21/14 (2013.01) *H04L 9/00* (2006.01)
(52) CPC특허분류
G06F 21/14 (2013.01)
H04L 9/002 (2013.01)
(21) 출원번호 10-2019-7029606
(22) 출원일자(국제) 2017년12월13일
심사청구일자 2019년10월08일
(85) 번역문제출일자 2019년10월08일
(65) 공개번호 10-2019-0122837
(43) 공개일자 2019년10월30일
(86) 국제출원번호 PCT/EP2017/082508
(87) 국제공개번호 WO 2018/162107
국제공개일자 2018년09월13일
(30) 우선권주장
10 2017 204 020.3 2017년03월10일 독일(DE)
(56) 선행기술조사문헌
US20140101458 A1*
KR101619458 B1*
KR101328012 B1
EP0900488 A1
*는 심사관에 의하여 인용된 문헌

(73) 특허권자
지멘스 악티엔게젤샤프트
독일 뮌헨 베르너-본-지멘스-슈트라쎄 1 (우: 80333)
(72) 발명자
즈완저, 요하네스
독일 85579 노이비베르크 하우프트슈트라쎄 6
(74) 대리인
특허법인 남앤남

전체 청구항 수 : 총 13 항 심사관 : 윤혜숙

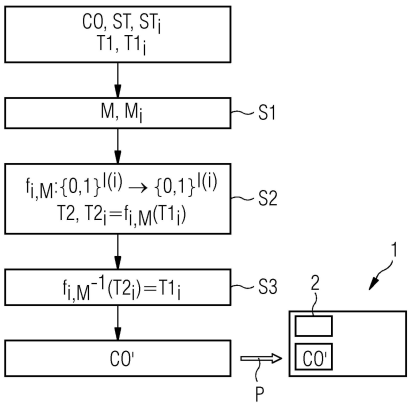
(54) 발명의 명칭 **프로그램 코드의 컴퓨터-지원 난독화를 위한 방법**

(57) 요 약

본 발명은 프로그램 코드(CO)의 컴퓨터-지원 난독화(computer-aided obfuscation)를 위한 방법에 관한 것으로, 복수의 계산 단계들(ST)이 프로그램 코드(CO)에서 구현되고, 복수의 계산 단계들(ST) 중 미리 결정된 계산 단계들이, 프로그램 코드(CO)의 실행과 함께 미리 결정된 순서로 리트리브되고, 미리 결정된 계산 단계들 중 적어도

(뒷면에 계속)

대 표 도 - 도1



일부는, 개개의 미리 정의된 계산 단계(ST_i)에 대해 요구되는 제1 테이블 값(tabular value)($T1_i$)을 제1 테이블(T1)로부터 판독하기 위해, 프로그램 코드(CO)에 저장된 그리고 복수의 디지털 제1 테이블 값들($T1_i$)로 구성된 개개의 제1 테이블(T1)이 액세스되는, 미리 정의된 계산 단계들(ST_i)이다. 프로그램 코드의 난독화의 부분으로서, 복수의 디지털 마스크 값들(M_i)에 의해 형성된 동적 마스크(M)가 사용되며, 임의의 미리 정의된 계산 단계의 경우, 제1 테이블(T1)로부터의 제1 테이블 값($T1_i$)을 제2 테이블 값($T2_i$)으로 대체하기 위해, 다른 마스크 값(M_i)이 사용된다. 게다가, 난독화될 프로그램 코드(CO)는, 개개의 미리 정의된 계산 단계(ST_i)에서의 프로그램 코드(CO)의 런타임 동안, 원래의 제1 테이블 값($T1_i$)으로의 제2 테이블 값($T2_i$)의 역계산(inverse calculation)이 수행되는 그러한 방식으로 적응된다. 본 발명에 따른 방법은, 프로그램 코드에 테이블 형태로 저장된 보호할 가치가 있는 정보(protection-worthy information)의 효율적인 난독화를 허용한다. 테이블 정보의 마스킹해제(demasking)는 프로그램 코드의 런타임 동안에 전체 프로그램 코드에 걸쳐 분산되며, 이로써, 인가되지 않은 공격자가 그 정보를 재구성하기가 더 어려워진다.

(52) CPC특허분류
G06F 2221/0748 (2013.01)
H04L 2209/16 (2013.01)

명 세 서

청구범위

청구항 1

프로그램 코드(CO)의 컴퓨터-지원 난독화(computer-aided obfuscation)를 위한 방법으로서,

상기 프로그램 코드(CO)에 다수의 컴퓨팅 단계들(ST)이 구현되고, 상기 프로그램 코드(CO)가 실행될 때 상기 다수의 컴퓨팅 단계들(ST) 중 미리 결정된 컴퓨팅 단계들이 미리 결정된 순서로 호출되고, 그리고 상기 미리 결정된 컴퓨팅 단계들 중 적어도 일부는 테이블-관련 컴퓨팅 단계들(ST_i)이고,

상기 테이블-관련 컴퓨팅 단계들(ST_i)의 각각에서, 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 대해 필요한 제1 테이블 값($T1_i$)을 제1 테이블(T1)로부터 판독하기 위해, 상기 프로그램 코드(CO)에 저장되고 그리고 다수의 디지털 제1 테이블 값들($T1_i$)을 포함하는 상기 제1 테이블(T1)이 액세스되고,

상기 프로그램 코드를 변경하기 위해,

다수의 디지털 마스크 값들(M_i)을 포함하는 동적 마스크(M)가 생성되는 단계 - 상기 마스크(M)의 적어도 일부의 마스크 값들(M_i)은 서로 상이하고 그리고 개개의 마스크 값(M_i)은 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 대해 유효함 -;

상기 제1 테이블(T1)의 각각의 제1 테이블 값($T1_i$)이 판독될 때 각각의 테이블-관련 컴퓨팅 단계(ST_i)에서 유효한 마스크 값(M_i)에 의해, 각각의 제1 테이블 값($T1_i$)이 디지털 제2 테이블 값($T2_i$)으로 변환되고, 그 결과로 상기 제1 테이블(T1) 대신에 상기 프로그램 코드(CO)에 저장되는 제2 테이블 값들($T2_i$)을 포함하는 제2 테이블(T2)이 획득되는 단계;

각각의 테이블-관련 컴퓨팅 단계(ST_i)에 대한 추가의 컴퓨팅 단계($f_{i,M}^{-1}$)가 상기 프로그램 코드(CO)에 구현되는 단계 - 상기 추가의 컴퓨팅 단계는 판독된 제2 테이블 값($T2_i$)을 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 필요한 상기 제1 테이블 값($T1_i$)으로 역변환함 - 가 수행되고,

상기 동적 마스크(M)를 생성하기 위해 초기 마스크 값 및 업데이트 단계들이 규정되고, 현재 유효한 마스크 값(M_i)에 하나 이상의 연속적인 업데이트 단계들을 적용함으로써 다음의(next) 테이블-관련 컴퓨팅 단계(ST_i)에 대한 유효한 마스크 값(M_i)이 계산되고, 상기 현재 유효한 마스크 값(M_i)이 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 존재하도록 상기 업데이트 단계들은 상기 프로그램 코드(CO)에도 구현되고, 상기 추가의 컴퓨팅 단계($f_{i,M}^{-1}$)는 개개의 테이블-관련 컴퓨팅 단계에서의 현재 유효한 마스크 값(M_i)에 의존하며, 상기 업데이트 단계들이 적용되는 마스크 값들(M_i) 중 적어도 일부의 마스크 값들에 대한 상기 업데이트 단계들은 상이하게 규정되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 2

제1항에 있어서,

상기 테이블-관련 컴퓨팅 단계들(ST_i) 각각에 상기 동적 마스크(M)를 생성하기 위한 업데이트 단계가 구현되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 3

제2항에 있어서,

상기 미리 결정된 컴퓨팅 단계들 중 상기 테이블-관련 컴퓨팅 단계들(ST_i)이 아닌 적어도 일부 컴퓨팅 단계들의 각각에도 상기 동적 마스크(M)를 생성하기 위한 업데이트 단계가 추가로 구현되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 4

제1항에 있어서,

상기 초기 마스크 값(M_i) 및/또는 상기 업데이트 단계들은 난수 발생기에 의해 규정되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 5

제1항에 있어서,

상기 제1 테이블 값들($T1_i$), 상기 제2 테이블 값들($T2_i$) 및 상기 디지털 마스크 값들(M_i)은 각각 비트 시퀀스(bit sequence)로 표현되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 6

제5항에 있어서,

각각의 제1 테이블 값($T1_i$)을 제2 테이블 값($T2_i$)으로 변환하는 것은, 상기 각각의 제1 테이블 값($T1_i$)의 비트 시퀀스와 상기 현재 유효한 마스크 값(M_i)의 비트 시퀀스 사이에 논리 연산들을 적용함으로써 수행되며, 상기 논리 연산들의 적용은 상기 제2 테이블 값($T2_i$)을 산출하는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 7

제6항에 있어서,

상기 논리 연산들은 상기 각각의 제1 테이블 값($T1_i$)의 비트 시퀀스와 상기 현재 유효한 마스크 값(M_i)의 비트 시퀀스의 상호 대응하는 비트들에 비트 단위로 적용되며, 상기 논리 연산들은 하나 이상의 OR 및/또는 XOR 및/또는 NOR 및/또는 XNOR 및/또는 AND 및/또는 NAND 연산들을 포함하는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 8

제7항에 있어서,

상기 현재 유효한 마스크 값(M_i)의 비트 시퀀스가 대응하는 제1 테이블 값($T1_i$)의 비트 시퀀스보다 더 짧은 경우, 상기 대응하는 제1 테이블 값($T1_i$)의 비트 시퀀스의 각 비트에 대해 상기 현재 유효한 마스크 값(M_i)의 비트 시퀀스의 대응 비트가 존재하도록, 상기 현재 유효한 마스크 값(M_i)의 원래의 비트 시퀀스의 비트들을 반복 사용함으로써, 상기 현재 유효한 마스크 값(M_i)의 원래의 비트 시퀀스가 연장되는 것을 특징으로 하는,

프로그램 코드(CO)의 컴퓨터-지원 난독화를 위한 방법.

청구항 9

제1항 내지 제8항 중 어느 한 항에 청구된 방법에 의해 난독화된 프로그램 코드(CO')를 실행하기 위한 방법으로

서,

상기 프로그램 코드(CO')의 개개의 테이블-관련 컴퓨팅 단계(ST_i)를 호출함으로써, 제2 테이블 값($T2_i$)이 제2 테이블($T2_i$)로부터 판독되고 그리고 추가의 컴퓨팅 단계($f_{i,M}^{-1}$)가 상기 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 대해 수행되며, 상기 추가의 컴퓨팅 단계는 상기 판독된 제2 테이블 값($T2_i$)을 상기 개개의 테이블-관련 컴퓨팅 단계(ST_i)에 필요한 제1 테이블 값($T1_i$)으로 역변환하는,

방법.

청구항 10

기술 시스템(technical system)으로서,

상기 기술 시스템(1)은, 제9항에 청구된 방법에 따라 프로그램 코드(CO')를 실행하도록 구성된 컴퓨터 수단(computer means)(2)을 포함하는 것을 특징으로 하는,

기술 시스템.

청구항 11

제10항에 있어서,

상기 기술 시스템(1)은 자동화 설비, 또는 자동화 설비의 컴포넌트, 또는 전력 생성 및/또는 전력 분배 시스템, 또는 전력 생성 및/또는 전력 분배 시스템의 컴포넌트, 또는 의료 기기인 것을 특징으로 하는,

기술 시스템.

청구항 12

프로그램 코드 섹션들이 저장된 컴퓨터-판독가능 저장 매체로서,

상기 프로그램 코드 섹션들은, 상기 프로그램 코드 섹션들이 컴퓨터 상에서 실행될 때, 제1항 내지 제8항 중 어느 한 항에 따른 방법을 수행하기 위한 것인,

컴퓨터-판독가능 저장 매체.

청구항 13

프로그램 코드 섹션들이 저장된 컴퓨터-판독가능 저장 매체로서,

상기 프로그램 코드 섹션들은, 상기 프로그램 코드 섹션들이 컴퓨터 상에서 실행될 때, 제9항에 따른 방법을 수행하기 위한 것인,

컴퓨터-판독가능 저장 매체.

발명의 설명

기 술 분 야

[0001] 본 발명은 프로그램 코드(program code)의 컴퓨터-지원 난독화(computer-aided obfuscation)를 위한 방법, 및 이러한 종류의 난독화된 프로그램 코드를 실행하기 위한 방법에 관한 것이다. 게다가, 본 발명은 기술 시스템(technical system), 컴퓨터 프로그램 제품(computer program product), 및 컴퓨터 프로그램을 포함한다.

배 경 기 술

[0002] 종종, 프로그램 코드로부터의 정보를 제3자들에 의한 인가되지 않은 액세스(access)로부터 보호할 필요가 있다. 이 경우, 대중적인 접근법은, 적절한 암호화 함수를 사용하여 정보를 암호화하는 것 또는 코드 패커(code packer)에 의해 프로그램 코드를 패키징(packaging)하는 것을 수반한다. 이 경우에서의 단점은, 정보가 다시 암호해독되거나 또는 프로그램 코드의 런타임(runtime)에서 코드가 다시 언패킹되는(unpacked) 것이어서, 보호할 가치가 있는 정보(information worthy of protection)가, 프로그램 실행 동안에 프로그램 메모리(program

memory)에서 평문(plain text)으로 완전히 이용가능하고, 따라서, 공격자가 적절한 기법들을 사용함으로써 이 정보를 메모리로부터 판독할 위험이 있다.

발명의 내용

[0003] 본 발명의 목적은, 프로그램 코드에서 보호할 가치가 있는 정보가 프로그램 코드의 런타임에서 제3자들에 의한 인가되지 않은 액세스로부터 매우 양호하게 보호되게 하는 컴퓨터-보호 방법(computer-protected method)을 제공하는 것이다.

[0004] 이 목적은 독립항들에 의해 달성된다. 본 발명의 개발예들은 종속항들로부터 정의된다.

[0005] 본 발명에 따른 방법은 프로그램 코드의 컴퓨터-지원 난독화(즉, 위장(disguise))를 위해 사용된다. 위장되도록 또는 난독화되도록 프로그램 코드에서 다수의 컴퓨팅 단계(computing step)들이 구현되며, 다수의 컴퓨팅 단계들 중 미리 결정된 컴퓨팅 단계들은 프로그램 코드가 실행될 때 미리 결정된 순서로 호출된다. 이 경우, 컴퓨팅 단계 및 또한 나중에 언급되는 업데이트 단계(update step)라는 용어는 광범위한 방식으로 이해되어야 한다. 컴퓨팅 단계는 반드시 단일 컴퓨팅 동작만을 포함할 필요는 없으며, 오히려, 가능하게는 조건들, 루프(loop)들, 네스트(nest)들 등을 갖는 다수의 컴퓨팅 동작들로 구성될 수 있다. 따라서, 하나의 컴퓨팅 단계 또는 업데이트 단계는 동작들의 시퀀스(sequence)를 캡슐화(encapsulate)할 수 있다.

[0006] 위장될 프로그램 코드의 미리 결정된 컴퓨팅 단계들은 정해진 컴퓨팅 단계(prescribed computing step)들을 포함한다. 미리 결정된 컴퓨팅 단계들은, 예컨대 정해진 컴퓨팅 단계들만을 포함할 수 있지만, 정해진 컴퓨팅 단계들 외에, 필요한 경우, 추가의 컴퓨팅 단계들이 또한 제공(provision)될 수 있다. 정해진 컴퓨팅 단계들은, 이러한 단계들 각각에서, 개개의 정해진 컴퓨팅 단계에 필요한 제1 테이블 값(table value)을 제1 테이블로부터 판독하기 위해, 프로그램 코드에 저장되고 그리고 다수의 디지털 제1 테이블 값(digital first table value)들을 포함하는 제1 테이블이 액세스된다는(accessed) 점에서, 구별된다. (존재하는 경우) 정해진 컴퓨팅 단계들이 아닌 모든 미리 결정된 컴퓨팅 단계들이, 그들을 호출하기 위한 규정된 미리 결정된 순서를 계속 갖고 있지만, 이러한 컴퓨팅 단계들은 제1 테이블에 액세스(access)하지 않는다.

[0007] 테이블이라는 용어는 광범위한 방식으로 이해되어야 한다. 테이블은, 적절한 커맨드(command)들을 통해 프로그램 코드에 의해 의도적으로 액세스될 수 있는 디지털 값들의 세트(set)이다. 또한, 테이블 값이라는 용어는 또한, 광범위한 방식으로 이해되어야 한다. 특히, 테이블 값은, 상응하게 호출된 컴퓨팅 단계에서 상이한 지점들에서 프로세싱되는(processed) 다수의 하위값들로 구성될 수 있다.

[0008] 본 발명에 따른 프로그램 코드 난독화는, 다수의 디지털 마스크 값(digital mask value)들을 포함하는 동적 마스크(dynamic mask)가 생성되는 것을 수반하며, 마스크의 적어도 일부의 마스크 값들 및 바람직하게는 마스크의 모든 마스크 값들은 서로 상이하며, 개개의 마스크 값은 개개의 정해진 컴퓨팅 단계에 대해 유효하다.

[0009] 본 발명에 따른 방법은, 제1 테이블의 각각의 제1 테이블 값이, 개개의 제1 테이블 값이 판독될 때 개개의 정해진 컴퓨팅 단계에서 유효한 마스크 값에 의해 디지털 제2 테이블 값으로 변환되는 것을 수반하며, 그 결과로 제1 테이블 대신에 프로그램 코드에 저장되는 제2 테이블 값들을 포함하는 제2 테이블이 획득된다. 다시 말해, 난독화된 프로그램 코드가 실행될 때, 더 이상 제1 테이블 값들이 아니라, 해당되는(applicable) 제2 테이블 값들이 액세스된다. 개개의 제1 테이블 값이 판독될 때, 개개의 정해진 컴퓨팅 단계에서 유효한 마스크 값은 또한, 아래에서 현재 유효한 마스크 값(currently valid mask value)으로 지칭된다.

[0010] 난독화된 프로그램 코드가, 난독화되지 않은 프로그램 코드와 동일한 결과들을 전달하도록, 난독화는, 추가의 컴퓨팅 단계가 각각의 정해진 컴퓨팅 단계에 대해 프로그램 코드에서 구현되는 것을 추가로 수반하며, 그 컴퓨팅 단계는, 해당되는 정해진 컴퓨팅 단계가 실행될 때, 판독된 제2 테이블 값을 개개의 정해진 컴퓨팅 단계에 필요한 제1 테이블 값으로 다시(back) 변환한다.

[0011] 본 발명에 따른 방법은, 동적 마스크가, 테이블 형태로 저장된 보호할 가치가 있는 정보를 매우 양호하게 위장하는 데 사용된다는 장점을 갖는다. 이 경우, 마스킹해제(demasking) 또는 공개(unveiling)의 동작들은 전체 프로그램 코드에서 다수의 컴퓨팅 단계들에 걸쳐 분산된다. 따라서, 난독화된 프로그램 코드의 런타임 동안, 위장된 정보의 재구성은 극심하게 방해받는다.

[0012] 특히 바람직한 실시예에서, 동적 마스크를 생성하기 위해 초기 마스크 값 및 업데이트 단계들이 규정되며, 다음 정해진 컴퓨팅 단계(next prescribed computing step)에 유효한 마스크 값은 하나 이상의 연속적인 업데이트 단계들을 현재 유효한 마스크 값에 적용함으로써 계산된다. 초기 마스크 값 및 업데이트 단계들은 또한, 현재 유

효한 마스크 값이 개개의 정해진 컴퓨팅 단계에서 존재하도록, 프로그램 코드에서 구현되며, 개개의 정해진 컴퓨팅 단계에서의 추가의 컴퓨팅 단계는 현재 유효한 마스크 값에 의존한다. 프로그램 코드에서의 업데이트 단계들의 구현의 결과로, 정해진 컴퓨팅 단계에서 제1 테이블 값을 재구성하려고 시도하는 공격자는 초기 마스크 값 및 이전의 업데이트 단계들의 지식을 필요로 한다. 이 지식이 프로그램 코드에 분산되어 있기 때문에, 프로그램 코드의 정보에 대한 특히 양호한 보호가 달성된다.

[0013] 다시(back) 원래의 제1 테이블 값들로의 변환을 방해하기 위해, 방금 설명된 실시예의 바람직한 변형은, 마스크 값들 중 적어도 일부, 특히 업데이트 단계들이 적용되는 모든 마스크 값들에 대한 업데이트 단계들이 상이하게 규정되는 것을 수반한다. 추가의 바람직한 변형에서, 정해진 컴퓨팅 단계들 각각에서 업데이트 단계가 제공(provision)되며, 바람직하게 업데이트 단계는 추가로, 정해진 컴퓨팅 단계들(존재하는 경우)이 아닌 미리 결정된 컴퓨팅 단계들 중 적어도 일부 각각에(특히 모든 미리 결정된 컴퓨팅 단계들에) 또한 제공된다.

[0014] 추가의 바람직한 변형에서, 초기 마스크 값 및 업데이트 단계들은 난수 발생기(random number generator)에 의해 규정된다. 이 방식으로, 이러한 값들 및 단계들의 매우 임의적인 규정이 달성되고, 따라서 프로그램 코드의 위장이 추가로 개선된다.

[0015] 추가의 바람직한 실시예에서, 제1 테이블 값들 및 제2 테이블 값들 및 또한 마스크 값들도 각각, 비트 시퀀스(bit sequence)로 표현된다. 바람직하게, 이 경우, 제2 테이블 값으로의 개개의 제1 테이블 값의 변환은, 제1 테이블 값의 비트 시퀀스와 현재 유효한 마스크 값의 비트 시퀀스 사이에 논리 연산들을 적용함으로써 실시된다. 이 경우, 논리 연산들의 적용은 제2 테이블 값을 전달한다.

[0016] 바람직하게, 상기 논리 연산들은 개개의 제1 테이블 값의 그리고 현재 유효한 마스크 값의 비트 시퀀스들의 상호 대응하는 비트들에 비트 단위로 적용되며, 논리 연산들은 바람직하게 하나 이상의 OR 및/또는 XOR 및/또는 NOR 및/또는 XNOR 및/또는 AND 및/또는 NAND 연산들을 포함한다. 따라서, 현재 유효한 마스크 값의 비트 시퀀스가 제1 테이블 값의 비트 시퀀스보다 더 긴 경우, 현재 유효한 마스크 값의 모든 비트들이 제1 테이블 값을 수정하는 데 사용되는 것은 아니다.

[0017] 현재 유효한 마스크 값의 비트 시퀀스가 개개의 제1 테이블 값의 비트 시퀀스보다 더 짧은 경우, 본 발명에 따른 방법의 바람직한 변형은, 현재 유효한 마스크 값의 비트 시퀀스의 대응하는 비트가 개개의 제1 테이블 값의 비트 시퀀스의 각각의 비트에 대해 존재하도록, 현재 유효한 마스크 값의 원래의 비트 시퀀스의 비트들을 반복적으로 사용함으로써 원래의 비트 시퀀스가 연장되는 것을 수반한다. 비트 시퀀스의 연장은 바람직하게, 원래의 비트 시퀀스가 1회 이상 반복되도록 실시된다. 이 방식으로, 마스크 값의 비트 길이가 해당되는 테이블 값의 비트 길이보다 더 짧은 경우에도, 해당되는(제1) 테이블 값의 양호한 위장이 달성된다.

[0018] 프로그램 코드를 난독화하기 위한 방법 외에, 본 발명은 또한, 난독화된 프로그램 코드를 실행하기 위한 방법에 관한 것이다. 이 경우, (난독화된) 프로그램 코드의 개개의 정해진 컴퓨팅 단계를 호출하는 것은, 제2 테이블 값이 제2 테이블로부터 판독되는 것 및 추가의 컴퓨팅 단계가 개개의 정해진 컴퓨팅 단계에 대해 수행되는 것을 초래하며, 그 추가의 컴퓨팅 단계는 판독된 제2 테이블 값을 개개의 정해진 컴퓨팅 단계에 필요한 제1 테이블 값으로 다시(back) 변환한다.

[0019] 난독화된 프로그램 코드가, 현재 유효한 마스크 값을 결정하기 위해 업데이트 단계들을 사용하는 실시예에 의해 생성된 경우, 난독화된 프로그램 코드의 실행은 또한, 난독화된 프로그램 코드에서 구현되는 업데이트 단계들이 수행되는 것을 수반한다.

[0020] 게다가, 본 발명은, 방금 설명된 방법에 따라 난독화된 프로그램 코드를 실행하도록 구성된 컴퓨터 수단을 포함하는 기술 시스템에 관한 것이다. 이 경우, 기술 시스템이라는 용어는 광범위한 방식으로 이해되어야 하며, 기술 시스템은 또한, 단일의 기술 기기일 수 있다. 이 경우, 난독화된 프로그램 코드는 상이한 기술 시스템들에 저장될 수 있다. 기술 시스템은, 예컨대 자동화 설비, 또는 자동화 설비의 컴포넌트(component), 또는 전력 생성 및/또는 전력 분배 시스템, 또는 전력 생성 및/또는 전력 분배 시스템의 컴포넌트, 또는 의료 기기일 수 있다.

[0021] 게다가, 본 발명은, 프로그램 코드의 컴퓨터-지원 난독화를 위한 본 발명에 따른 방법을 수행하기 위한, 또는 난독화된 프로그램 코드를 실행하기 위한 본 발명에 따른 방법을 수행하기 위한, 또는 프로그램 코드 섹션(program code section)들이 컴퓨터 상에서 실행될 때, 이러한 방법들의 바람직한 변형들을 수행하기 위한, 기계-판독가능 캐리어(machine-readable carrier) 상에 저장된, 프로그램 코드 섹션들을 갖는 컴퓨터 프로그램 제품에 관한 것이다.

[0022] 게다가, 본 발명은, 프로그램 코드의 컴퓨터-지원 난독화를 위한 본 발명에 따른 방법을 수행하기 위한, 또는 난독화된 프로그램 코드를 실행하기 위한 본 발명에 따른 방법을 수행하기 위한, 또는 프로그램 코드 섹션들이 컴퓨터 상에서 실행될 때, 이러한 방법들의 바람직한 변형들을 수행하기 위한, 프로그램 코드 섹션들을 갖는 컴퓨터 프로그램에 관한 것이다.

[0023] 상기 컴퓨터 프로그램 제품 또는 컴퓨터 프로그램이, 프로그램 코드를 난독화하기 위한 방법을 수행하는 데 사용되는 경우, 난독화는 해당되는 프로그램 코드 섹션들에 의해 야기된다. 따라서, 프로그램 코드 섹션들은 난독화될 프로그램 코드가 아니다.

[0024] 컴퓨터 프로그램 제품 또는 컴퓨터 프로그램이, 난독화된 프로그램 코드를 실행하는 데 사용되는 경우, 프로그램 코드 섹션들은 난독화된 프로그램 코드에 대응한다.

도면의 간단한 설명

[0025] 본 발명의 예시적인 실시예는 첨부된 도 1을 참조하여 아래에서 상세하게 설명된다. 도 1은 프로그램 코드를 난독화하기 위한, 본 발명에 따른 방법의 실시예에 대한 흐름도를 도시한다.

발명을 실시하기 위한 구체적인 내용

[0026] 도 1에 따르면, 방법에 대한 시작점은, 본 발명의 범위 내에서 위장될 필요가 있는 프로그램 코드(CO)이다. 이 프로그램 코드에는 다수의 컴퓨팅 단계들이 제공되며, 그 다수의 컴퓨팅 단계들은 도 1에서 ST로 표시된다. 이 경우, 이러한 컴퓨팅 단계들(ST_i(i=1, …, n)) 중 일부는 제1 테이블(T1)에 액세스하고, 개개의 컴퓨팅 단계는 테이블로부터 해당되는 엔트리(applicable entry)($T1_i$)를 판독한다. 인덱스(index) i는, 컴퓨팅 단계들(ST_i)이 연속적으로 실행되는 미리 결정된 순서를 규정하는 데 사용된다. 이 경우, 컴퓨팅 단계들(ST_i)은 청구항 제1항의 맥락 내에서, 정해진 컴퓨팅 단계들에 대응한다. 컴퓨팅 단계들(ST_i) 사이에서, 프로그램 코드(CO)의 또 다른 추가의 컴퓨팅 단계들이 수행되는 것이 또한 가능하지만, 이러한 추가의 컴퓨팅 단계들은 테이블(T1)에 액세스(accessing)하지 않는다.

[0027] 따라서, 프로그램 코드(CO)에 따라, 제1 테이블(T1)의 특정 엔트리($T1_i$)는 컴퓨팅 단계(ST_i)에서만 액세스된다. 이 경우, 개개의 테이블 엔트리는 해당되는 컴퓨팅 단계(applicable computing step)에서 프로세싱되는 비트 시퀀스이다. 이 경우, 개별적인 테이블 엔트리들의 비트 시퀀스들은 길이 l(i)을 갖는다. 이러한 길이들은 상이한 테이블 엔트리들에 대해 상이한 크기일 수 있다.

[0028] 프로그램 코드(CO)는, 런타임에서 컴퓨팅 단계들(ST)을 적용함으로써, 알려진 입력으로부터 특정 출력을 생성한다. 이 경우, 근본적으로, 컴퓨팅 단계들(ST)은 임의의 순서로 수행될 수 있지만, 특정 컴퓨팅 단계들(ST_i)은 항상 서로 동일한 순서로, 그리고 입력에 상관없이 항상 정확하게 한 번 수행된다. 개개의 컴퓨팅 단계(ST_i)는 항상 단일 테이블 엔트리($T1_i$)에만 액세스하지만, 컴퓨팅 단계는, 그 컴퓨팅 단계가 수행될 때, 이 테이블 엔트리에 원하는 만큼 여러 번 액세스하도록 허용된다.

[0029] 해당되는 테이블 엔트리들($T1_i$)을 갖는 위장되지 않은 테이블(T1)은 보호할 가치가 있는 정보를 포함하며, 인가되지 않은 제3자가 프로그램 코드로부터 테이블 엔트리들을 재구성할 수 없도록 또는 매우 큰 노력으로만 그 테이블 엔트리들을 재구성할 수 있도록, 프로그램 코드(CO)를 난독화하는 것이, 아래에서 설명되는 실시예의 목표이다. 전체 테이블(T1)에 암호화 기능을 1회 적용하는 것과 대조적으로, 위장은 전체 프로그램 코드에 걸쳐 분산되는 방식으로 제공되며, 이는 다시(back) 원래의 테이블 값들로의 변환을 극심하게 방해한다.

[0030] 도 1의 단계(S1)에 따라, 프로그램 코드(CO)를 난독화하기 위해, 마스크 엔트리들(M_i(i=1, …, n))을 갖는 동적 마스크(M)가 가장 먼저 결정된다. 따라서, 각각의 정해진 컴퓨팅 단계(ST_i)에 대한 대응하는 마스크 값(M_i)이 존재한다. 이 마스크 값(M_i)은 컴퓨팅 단계(ST_i)에 대해서만 유효하다. 마스크 값들의 인덱싱(indexing)은 컴퓨팅 단계들(ST_i)의 수행에 대응하는 순서를 규정한다. 개별적인 마스크 값들(M_i)은 각각 특정 길이를 갖는 비트 시퀀스를 나타내며, 이 길이는 여기서 설명된 실시예에서 마스크 값들(M_i)에 걸쳐 동일하게 유지되지만, 반드시 그럴 필요는 없다.

[0031] 개별적인 마스크 값들(M_i)은, 임의적인 초기 마스크 값으로부터 시작하여 가장 최근에 업데이트된(updated) 마스크 값까지 연속적으로 적용되는 일련의 업데이트 단계들에 의해 결정된다. 이 경우, 업데이트 단계라는 용어는 광범위한 방식으로 이해되어야 한다. 특히, 업데이트 단계는 단일 동작뿐만 아니라, 필요한 경우, 다수의 동작들도 포함할 수 있다. 여기서 설명되는 실시예에서, 업데이트 단계들은 임의적으로 규정되었으며, 따라서 적어도 부분적으로 서로 상이하다. 그럼에도 불구하고, 개별적인 업데이트 단계들은 확실하게 정해지며, 그 안에 포함된 동작들은 고정된 순서로 수행된다. 따라서, 단계(S1)에 따라, 임의적인 마스크 할당들 또는 마스크 값들(M_i)을 갖는 마스크(M)가 생성된다. 이 경우, 바람직하게, 마스크의 초기 값을 규정하고 업데이트 단계들을 규정하기 위해, 난수 발생기가 사용된다.

[0032] 그 다음으로, 단계(S2)에서, 개개의 컴퓨팅 단계(ST_i)에서 판독된 테이블 엔트리($T1_i$)는, 해당되는 컴퓨팅 단계(ST_i)에서 유효한 마스크 값(M_i)에 의존하는 함수($f_{i,M}$)에 의해 변경된다. 이 경우, 함수($f_{i,M}$)는 임의의 방식으로 규정되었을 수 있는데; 함수가 개개의 마스크 값(M_i)에 의존하고 그리고 테이블 엔트리($T1_i$)를 제2 테이블(T2)의 새로운 테이블 엔트리($T2_i$)에 전단사 방식으로 맵핑(bijectively map)하는 것이 단지 중요하다. 도 1로부터 확인될 수 있는 바와 같이, 이 예시적인 실시예의 함수($f_{i,M}$)가 맵핑(mapping)되며, 이는 테이블 엔트리($T1_i$)의 비트 시퀀스를, 동일한 길이를 갖는 다른 비트 시퀀스에 맵핑하고, 이 다른 비트 시퀀스는 새로운 테이블 엔트리($T2_i$)에 대응한다.

[0033] 수정된 테이블 엔트리들($T2_i$)이 모든 컴퓨팅 단계들(ST_i)에 대해 생성된 후에, 원래의 테이블 엔트리들($T1_i$)은 이러한 새로운 테이블 엔트리들($T2_i$)로 대체되는데, 이는 원래의 테이블 엔트리들이 공격자에 의해 프로그램 코드(CO)로부터 용이하게 판독될 수 없다는 것을 의미한다. 테이블 엔트리들의 수정으로 인해, 프로그램 코드(CO)는 또한, 판독된 테이블 엔트리($T2_i$)가 해당되는 컴퓨팅 단계(ST_i)의 원래의 테이블 엔트리($T1_i$)로 변환된다는 의미에서, 변경될 필요가 있다. 이는 도 1에 따라 단계(S3)에서 달성된다. 이 단계에서, 초기 마스크 값은 프로그램 코드의 시작 시에 저장되고, 위의 업데이트 단계들이 또한 프로그램 코드에서 구현되며, 개개의 컴퓨팅 단계(ST_i)를 호출하는 것은, 여기서 설명된 실시예에서 현재 유효한 마스크 값을 결정하기 위해, 해당되는 업데이트 단계가 수행되는 것을 초래한다. 게다가, 각각의 컴퓨팅 단계에서 추가의 단계가 구현되는데, 이 추가의 단계는, 현재 유효한 마스크 값(M_i)에 의존하는 역함수($f_{i,M}^{-1}$)에 의해, 판독된 테이블 엔트리($T2_i$)를 원래의 테이블 엔트리($T1_i$)로 변환한다.

[0034] 따라서, 단계들(S1 내지 S3)의 결과로, 원래의 테이블(T1)이 위장된, 난독화된 프로그램 코드(CO')가 획득된다. 그러나, 원래의 테이블 값들을 획득하기 위한 역변환 단계(back-conversion step)는 난독화되지 않은 프로그램 코드(CO)와 동일한 컴퓨팅 결과를 획득한다. 난독화된 프로그램 코드(CO')는 그 후에, 임의의 기술 기기에 저장되어 실행될 수 있다. 이는 도 1에서 화살표(P)로 표시된다. 이 경우, 난독화된 프로그램 코드(CO')는 기술 기기(1)에 저장되며, 그 기술 기기(1)는 난독화된 프로그램 코드(CO')를 실행하는 데 사용되는 컴퓨터 수단(2)을 갖는다.

[0035] 프로그램 코드에 대해 위에서 설명된 난독화는 다양한 기술 분야들에서 사용될 수 있다. 이 경우, 특히, SCADA 시스템들, 프로그램가능 로직 제어기(PLC; programmable logic controller)들, 모션 제어 시스템(motion control system)들, 자동화 소프트웨어(automation software), 컴퓨터 단층촬영 스캐너(computed tomography scanner)들, 스마트그리드(SmartGrid)들 등의 프로그램 코드가 변경될 수 있다. 이 경우, 임의의 알고리즘(algorithm)들, 예컨대 개인용 컴퓨터(PC; personal computer)들 상의 프로그램들 또는 기기들 상의 펌웨어(firmware)가 난독화되는 것이 가능하다. 프로그램들은 임의의 태스크(task)들을 떠맡을 수 있는데, 예컨대 개루프(open-loop) 및/또는 폐루프(closed-loop) 제어 알고리즘들 또는 뉴럴 네트워크(neural network)들에 기반하는 알고리즘들일 수 있다.

[0036] 일반적으로, 본 발명에 따른 난독화에 의해 달성되는 효과는, 테이블 형태로 저장되고 보호할 가치가 있는 데이터(data)가 하나의 소프트웨어로 위장된다는 것이다. 보호할 가치가 있는 데이터는, 예컨대 암호화 정보(cryptographic information), 또는 라이센스 검사(license check)들에 대해 중요한 정보일 수 있다.

[0037] 본 발명에 따른 방법에 대해 위에서 설명된 실시예는 일련의 장점들을 갖는다. 특히, 테이블을 마스킹(masking)하는 것은, 공격자가 더 이상, 이 테이블로부터 어떤 종류의 정보도 추출할 수 없다는 것을 의미한다.

테이블의 마스킹된 형식(masked form)을 유용하게 해석할 수 있기 위해서는, 마스크의 초기화 값 및 마스크에 대한 전체적인 연속적인 업데이트 단계들에 관한 추가의 지식이 대신 필요하다.

[0038] 일회성 암호해독 또는 언패킹(unpacking) 동작들과 대조적으로, 테이블을 마스킹해제(demasking)하는 단계들은, 프로그램 코드의 격리된 지점에서 발생하기보다는, 마스킹해제는 전체 프로그램 코드에 걸쳐 분산된다. 따라서, 공격자는 테이블의 원래의 값들에 관한 결론들을 내리기 위해서는 전체 프로그램 코드를 분석할 필요가 있다. 반대로, 테이블의 일회성 암호화의 경우, 공격자는 암호해독의 지식이 주어지면 테이블 엔트리들의 평문에 즉시 액세스할 수 있다.

[0039] 본 발명에 따른 난독화는 추가로, 현재 단계에서 사용된 테이블 엔트리보다 더 많은 것이, 프로그램 코드의 실행 시에 언제든지 메모리 상태, 즉, 특히 현재 마스크 값에 기반하여 용이하게 추론되는 것을 허용하지 않는다. 마스크 값이 다음 업데이트가 되자마자, 이전의 테이블 엔트리를 추론하는 것은 더 이상 가능하지 않다.

[0040] 본 발명에 따른 난독화 방법은 추가로, 리버스 엔지니어링(reverse engineering)의 추가의 기법들, 이를테면, 예컨대, 안티 디버그 조치(anti debug measure)들 또는 자기-수정 코드(self-modifying code)들의 사용과 매우 양호하게 조합될 수 있다. 이 방식으로, 공격자가 원래의 테이블 값들을 추출하기 위해 마스킹해제 동작들의 전체 시퀀스를 재구성하는 것은 훨씬 더 어려워진다.

도면

도면1